AD-A094 008 HONEYWELL SYSTEMS AND RESEARCH CENTER MINNEAPOLIS MN F/G 17/8
ADVANCED TARGET TRACKER CONCEPTS.(U)
APR 80 P M NARENDRA, B L WESTOVER DAAK70-79-C-0150
UNCLASSIFIED 80SRC47 NL

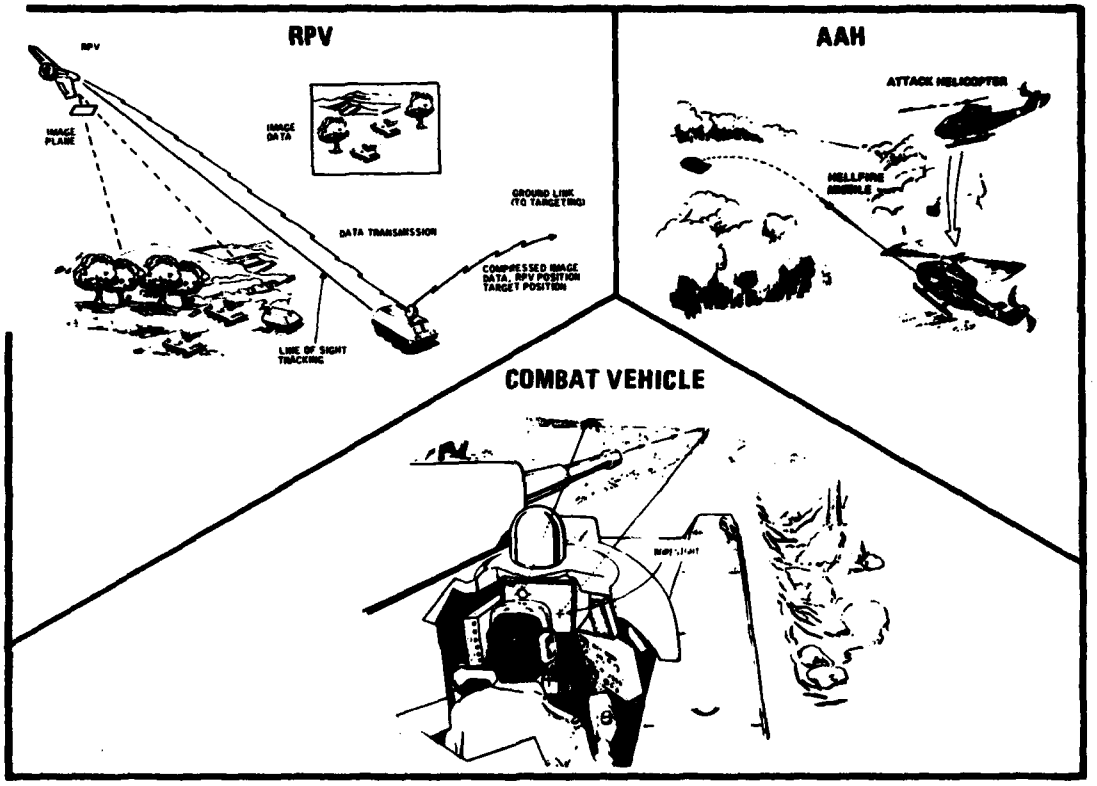
1 OF 1
AD A094008

END
DATE FILMED
2-81
DTIC

LEVEL III

3

ADVANCED TARGET TRACKING CONCEPTS



SECOND QUARTERLY PROGRESS REPORT

29 DECEMBER 1979 to 30 MARCH 1980

Prepared for
UNITED STATES ARMY
Night Vision and Electro-Optics Laboratory
Fort Belvoir, Virginia 22060

Honeywell

SYSTEMS & RESEARCH CENTER
2600 RIDGWAY PARKWAY MINNEAPOLIS, MINNESOTA 55413

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A094008 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADVANCED TARGET TRACKER CONCEPTS. AD-A083 710	5. TYPE OF REPORT & PERIOD COVERED Quarterly Progress Report. 29 Dec 79 - 30 Mar 80,
	6. PERFORMING ORG. REPORT NUMBER 80SRC47
7. AUTHOR(s) P. M. Narendra and B. L. Westover	8. CONTRACT OR GRANT NUMBER(s) DAAK70-79-C-0150
9. PERFORMING ORGANIZATION NAME AND ADDRESS Honeywell Systems & Research Center 2600 Ridgway Parkway Minneapolis, Minnesota 55413	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 90
11. CONTROLLING OFFICE NAME AND ADDRESS Night Vision and Electro-Optics Laboratory Fort Belvoir, Virginia 22060	12. REPORT DATE Apr 1980 / 13. NUMBER OF PAGES 91
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified / 15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

The project monitor at NV&EOL is CPT Benjamin Reischer.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pattern recognition	Target cueing	Scene analysis
Multiple targets	Target screening	Artificial intelligence
Target tracking	Image processing	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Conventional target tracking approaches rely on numerical correlation over successive frames on a window around the target. They are therefore sensitive to partial obscuration and changes in target and background appearances. Furthermore, multiple-target tracking requires replication of the hardware.

In this report, we present the development of a multiple-target tracking approach based on a dynamic scene model derived from the analysis of a time

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

sequence of imagery. Simulation results demonstrate multiple-target tracking in cluttered backgrounds and in imagery from fast-moving platforms. The approach can be implemented as an integral part of the Honeywell target screener system.

CONTENTS

CONTENTS (concluded)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (continued)

LIST OF ILLUSTRATIONS (concluded)

SECTION 1

INTRODUCTION

This is the Second Quarterly Progress Report on "Advanced Target Tracker Concepts," NV&EOL Contract No. DAAK70-79-C-0150. It reports the results of the work performed between 28 December 1979 and 30 March 1980.

Tracking targets in video from TV and FLIR sensors is essential for fire control in weapon systems using electro-optical target acquisition. Figure 1 shows typical Army applications: a remotely piloted vehicle (RPV), an advanced attack helicopter (AAH), and a combat vehicle (CV). Target tracking in these applications yields the target position for accurate pointing of a laser designator for a smart munition, such as Hellfire and Copperhead, or for fire control of conventional weapons.

Currently fielded trackers rely on numerical correlation over successive frames on a window around the target to be tracked. Several variations of the basic correlation scheme exist, and a detailed survey can be found in "Assessment of Target Tracking Techniques."[1] Conventional trackers are capable of tracking a manually acquired single target in relatively clutter-free backgrounds. However, target tracking requirements in the increasingly sophisticated weapon systems have grown beyond the capabilities of the current correlation trackers.[1]

---

[1] Reischer, B., "Assessment of Target Tracking Techniques," Proceedings of SPIE, Vol. 178, Smart Sensors, pp. 67-71, 1979.

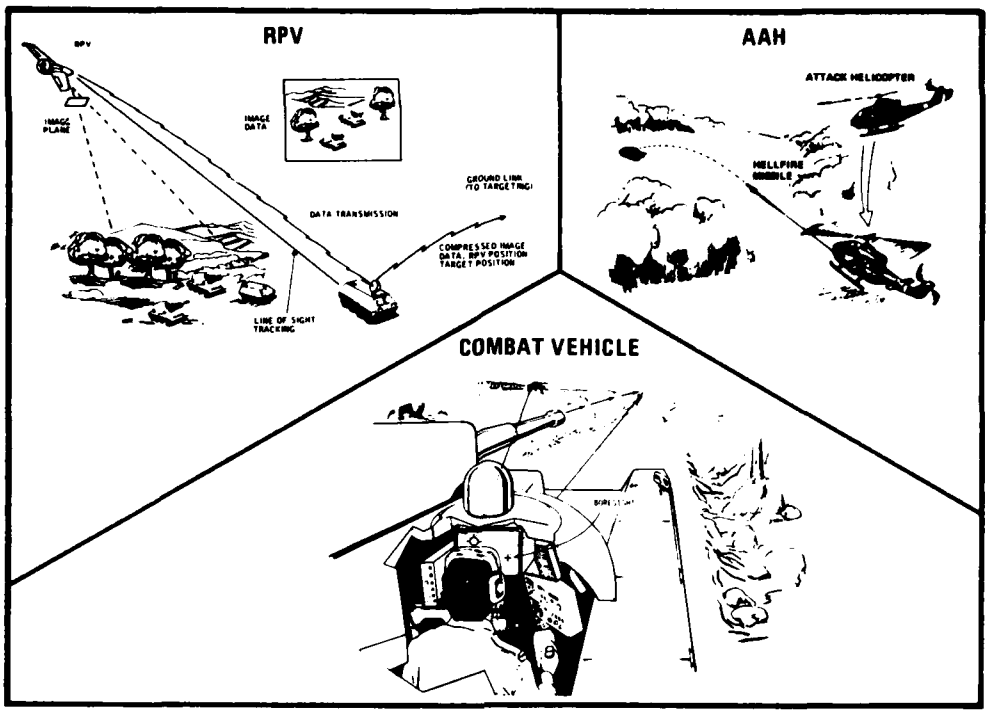


Figure 1. Typical Army Scenarios Which Require Advanced Multiple-Target Tracking Through High Clutter

Among these requirements are:

- Automatic target detection (acquisition), recognition, and prioritization
- Simultaneous tracking of multiple targets in the presence of clutter, obscuration, and low contrast
- Critical aimpoint selection

In this program Honeywell Systems and Research Center is developing an advanced target tracker approach, based on dynamic scene analysis, which will satisfy these requirements. This approach integrates the

target screening and tracking functions which can provide automatic acquisition and multiple-target tracking through low signal-to-noise and high clutter conditions. This is done with minimal additional hardware to a target screener.

Figure 2 is an overview block diagram of the basic approach which builds upon the scene analysis functions performed by the target screener to perform the advanced tracking function. The basic premise is very simple: the target screener segments and classifies significant objects (targets and clutter) in real time on a frame-by-frame basis. The symbolic descriptions of the objects in each frame are used to find the corresponding objects in previous frames encompassing the history of the scene. Once the corresponding object matches are made, the scene model, which

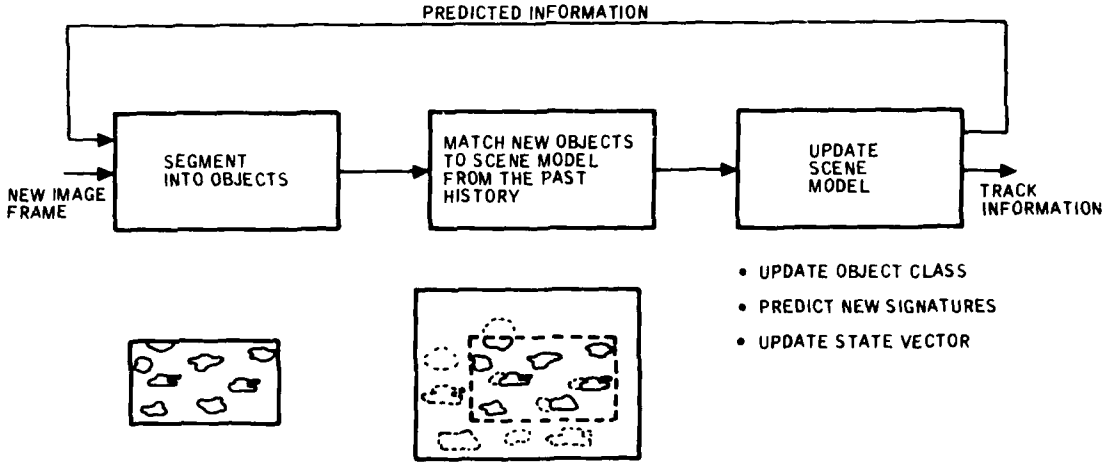


Figure 2. Overview of the Advanced Target-Tracking Approach

includes the sensor and object dynamics as well as the target classes, is updated. Because we are keeping track of the positions of all the objects in the scene (targets and clutter), we can predict impending occlusion and future target/background signatures. Multiple-target tracking, of course, comes free. The scene model, based on the past history of the scene, can extend beyond the current field of view. This allows reacquisition and tracking of targets which wander in and out of the field of view because of sensor platform motion.

A complete block diagram of the major functions necessary to implement the advanced target-tracker concept is shown in Figure 3.

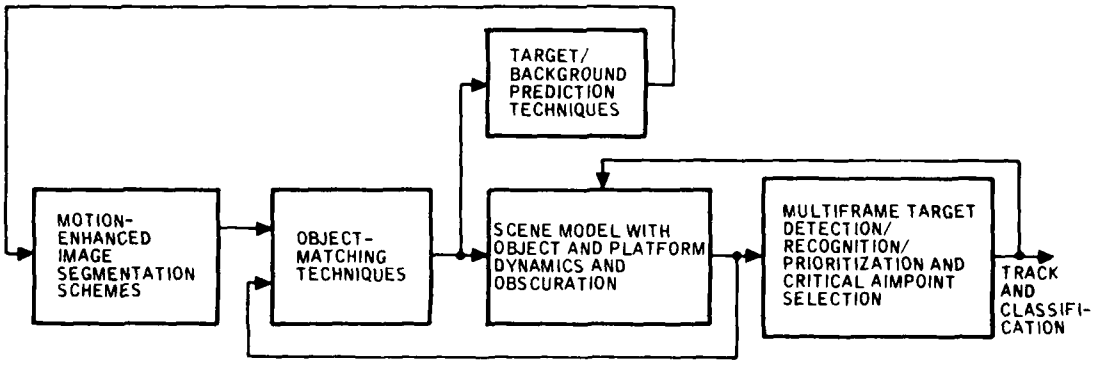


Figure 3. Advanced Target Tracker Program Overview with the Key Functions

These functions represent the major thrusts of the current program. They are:

- Efficient motion-enhanced scene segmentation schemes
- Object-matching techniques capable of precise matching of objects in the new frame to the scene model derived from previous frames
- A scene model capable of characterizing object and platform dynamics, target/background signatures, and object occlusion
- Target/background signature prediction techniques to improve the probability of target acquisition in low signal-to-noise ratios
- Advanced target detection/recognition/prioritization and critical aimpoint selection algorithms which can exploit the dynamic multiframe information

Each of these functions is introduced briefly below.

MOTION-ENHANCED SEGMENTATION SCHEMES

Object extraction (segmentation) in the integrated tracker/screener application is unique in that each frame is being analyzed in the context of the previous frames. However, conventional techniques for image segmentation do not use information from the previous frames to segment objects in the current frame. The current program uses the Honeywell Prototype Automatic Target Screener (PATS) segmentation algorithm as the baseline segmentation approach. This segmentation technique will

be modified to incorporate the a priori predicted information on object/background signatures for more optimal segmentation. This effort will be directed at incorporating the interframe knowledge of the target shape and intensity signatures, as well as background characteristics expected at various locations in the frame as predicted by the scene model below.

OBJECT-MATCHING TECHNIQUES

The key to successful tracking of multiple targets in our approach depends on precise matching of segmented objects in the current frame with the scene model derived from previous frames. This allows the precise tracking of the object positions for laser designation or for hand-off to other subsystems. Key issues in object-matching techniques are unambiguous matching in the presence of occlusion, segmentation differences due to noise, and computational efficiency of the algorithm.

SCENE MODEL

The scene model is a collection of information from previous frames against which the new frame can be compared. It consists of the object shapes and positions from previous frames, the object dynamics (object positions and velocities), and the sensor/platform motion dynamics (position and velocity). In addition, the scene model must be capable of predicting occlusion and signature change of a target as it approaches occluding objects.

TARGET/BACKGROUND SIGNATURE PREDICTION TECHNIQUES

The purpose of this effort is to use the multiframe information on the target position and dynamics to predict the target shape, intensity signatures and position, and background characteristics expected at various locations in the frame. This information is used by the motion-enhanced segmentation scheme to increase the probability of target acquisition in the presence of low signal-to-noise ratios and high clutter.

ADVANCED ALGORITHMS FOR TARGET DETECTION/RECOGNITION/ PRIORITIZATION AND CRITICAL AIMPOINT SELECTION

Detection/recognition functions are performed in current target screeners on a frame-by-frame basis. The purpose of this task is to use the multiframe information to improve the performance of these functions in the integrated system. This improvement will be brought about in two ways: 1) by accumulating multiframe decisions of corresponding objects to improve the classification accuracy over single-frame analysis; and 2) by taking advantage of the fact that moving objects will, in general, be targets. Thus, the problem of target recognition can be improved by a moving-target detection algorithm. Critical aimpoint selection is an important function required in terminal homing munitions, and its implementation with syntactic techniques will be addressed in subsequent reporting periods.

SUMMARY OF PROGRESS

Several accomplishments toward the program objectives were made during this reporting period:

- A noniterative, fast silhouette-matching algorithm was developed in addition to the iterative algorithm developed in the last reporting period. This algorithm is independent of the starting point of the search and finds the best match of extracted object outlines without multiple iterations.
- Analysis of both fast silhouette-matching algorithms has started in order to characterize their performance.
- A flexible scene model data structure was developed to represent objects from multiple frames and their relationships to one another. It can represent several tracking problems such as occlusion, missegmentation, multiple component objects, etc. It will prove to be a considerable aid in the resolution of these problems.
- The system simulation was modified to incorporate the new data structure above.
- The PATS simulation software converted to the EIKON data handling system was transferred and installed at the IBM 360 based image processing facility at NV&EOL. An extensive documentation and user manual were prepared (see Appendix A).

REPORT ORGANIZATION

The remaining sections of the report are organized as follows:

- Object-Matching Algorithms
- Scene Model
- System Simulation
- PATS Simulation Transfer
- Plans for the Next Reporting Period

SECTION 2

OBJECT-MATCHING SCHEMES

Object matching is performed on the output of object segmentation. Its purpose is to find the positions of corresponding objects in successive frames. It is, therefore, the key to tracking the object positions as the sensor and the targets move from one frame to the next. Object matching not only finds the positions of the moving targets in successive frames but also identifies corresponding stationary (clutter) objects in the scene. The positions of these corresponding stationary objects are input to the scene (sensor/platform) dynamics model for computing the platform motion.

The key issues to be addressed in the development of successful object-matching algorithms are:

- Occlusion
- Inconsistent segmentation

These issues were addressed in detail in the first quarterly report.[2]

---

[2] P. M. Narendra and B. L. Westover, "Advanced Target Tracker Concepts," First Quarterly Report, Honeywell Systems and Research Center, Minneapolis, Minnesota, January 1980.

The principal effect of object occlusion (partial or total) is that the object shape descriptors change, making it difficult to match objects in successive frames. For example, when a target goes behind concealing background, the leading edge of the target disappears. Inconsistent segmentation usually results from poor signal-to-noise ratio and segmentation algorithm anomalies. For example, objects extracted in one frame may not appear in the subsequent frames; an object extracted as one segment in one frame may appear as multiple segments in the subsequent frames or vice versa. The outlines of the segments extracted may change shape drastically because of change in target/background contrast from one frame to the next.

In the previous reporting period, two object-matching algorithms were developed. One is the simple feature-based, object-matching technique which finds corresponding objects based on simply derived object descriptors such as contrast, shape, etc. It succeeds in finding initial matches of corresponding objects with consistent segmentations. To handle inconsistent segmentations and to obtain precise positions of objects in successive frames, a fast iterative, silhouette-matching algorithm was developed. This algorithm works on the segmented outlines of the objects and rapidly converges to a precise registration of objects in successive frames. The nature of this algorithm allows it to handle inconsistent segmentations which result in one-to-one, one-to-many, many-to-one, and many-to-many object matches. In this reporting period, a noniterative silhouette-matching algorithm was developed. This algorithm, although similar to the fast silhouette-matching algorithm, requires only one iteration to find a precise registration of the object outlines.

NONITERATIVE SILHOUETTE-MATCHING ALGORITHM

The iterative, fast silhouette-matching algorithm described in the last quarterly progress report, iteratively shifts an object outline from the old frame until a precise match with an object outline in the current frame is found. The amount of the shift is determined by a one-dimensional histogram of the differences in the positions of the edge pixels in the old frame and current frame. The peak in this histogram gives the desired shift. The histograms are first computed for horizontal differences. Then the required shift is added to the old object outline. In a similar fashion, the vertical shift required to align the objects is found and applied to the old object outline. This procedure is iterated until no translation of the old object outline in either the horizontal or vertical direction will improve the registration.

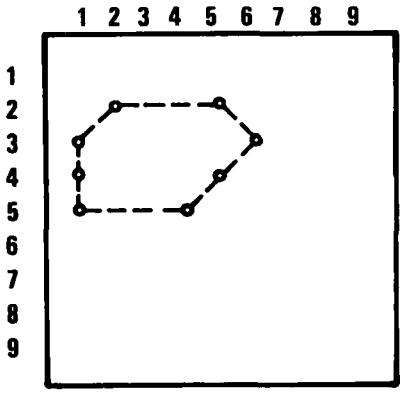
This algorithm shares a few issues with all iterative algorithms. First, what is the impact of the convergence properties of this iterative technique on real-time implementation? Second, is the algorithm sensitive to initial starting points? The noniterative algorithm removes both of these potential problems. It will find the best match of two silhouettes in one iteration, independent of the starting location. The algorithm determines both the best horizontal shift and best vertical shift of the old object outline simultaneously. The technique is similar to the fast silhouette-matching algorithm (FSMA) developed last reporting period, but it uses two-dimensional histograms instead of one-dimensional histograms. It is described in the following paragraphs.

The FSMA computes the best one-dimensional (horizontal or vertical) shift by computing a one-dimensional histogram of differences in the locations of edge pixels.[2] In a similar fashion, the two-dimensional histogram algorithm computes the best two-dimensional shift (horizontal and vertical) by computing a two-dimensional histogram of edge pixel differences. For every pair of points, from the object model outline and from the new object outline, we can find a translation which exactly aligns the two points. If the point in the object model has line and column numbers given by $\left(L^{(o)}, C^{(o)}\right)$ and the point in the new frame is determined by $\left(L^{(n)}, C^{(n)}\right)$, then the translation required to correctly align these two points is given by $\left(L^{(n)} - L^{(o)}, C^{(n)} - C^{(o)}\right)$. If we create a histogram of these differences for all pairs of points in the two outlines, then the peak in this histogram will yield the translation which exactly aligns the most pixels in the two outlines.

This technique is illustrated in Figure 4. In this figure we see an object silhouette from the scene model from the previous frame and a segmented object outline from a new frame. Note how the segmented object does not exactly fit the object model. The two-dimensional, silhouette-matching algorithm aligns the corresponding edges of the two outlines.

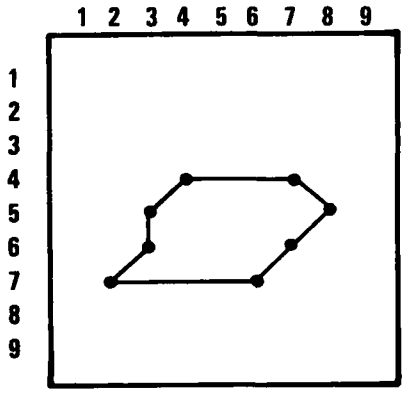
In Figure 4b the differences of the edge pixel locations for each pair of points in the left and right edges have been computed and histogrammed. Both left and right edge difference histograms have peaks at (2, 2). This peak equals the translation of the object model in the current frame. The result of applying this translation to the object model is shown in Figure 4c. Note the exact alignment of the right edge and the alignment of corresponding parts of the left edge.

OBJECT OUTLINE FROM SCENE MODEL



LEFT EDGE	RIGHT EDGE
(2,2)	(2,5)
(3,1)	(3,6)
(4,1)	(4,5)
(5,1)	(5,4)

SEGMENTED OBJECT OUTLINE FROM CURRENT FRAME



LEFT EDGE	RIGHT EDGE
(4,4)	(4,7)
(5,3)	(5,8)
(6,3)	(6,7)
(7,2)	(7,6)

Figure 4a. Object Model and Segmented Object Outlines with Endpoint Coordinates (Note that the segmented object does not exactly fit the object model.)

LEFT EDGE DIFFERENCES

(4,4) - (2,2) = (2,2)
(4,4) - (3,1) = (1,3)
(4,4) - (4,1) = (0,3)
(4,4) - (5,1) = (-1,3)
(5,3) - (2,2) = (3,1)
. . .
. . .
. . .
(7,2) - (4,1) = (3,1)
(7,2) - (5,1) = (2,1)

RIGHT EDGE DIFFERENCES

(4,7) - (2,5) = (2,2)
(4,7) - (3,6) = (1,1)
(4,7) - (4,5) = (0,2)
(4,7) - (5,4) = (-1,3)
(5,8) - (2,5) = (2,3)
. . .
. . .
. . .
(7,6) - (4,5) = (3,1)
(7,6) - (5,4) = (2,2)

HISTOGRAM OF LEFT EDGE DIFFERENCES

COLUMN SHIFT

LINE SHIFT	-2	-1	0	1	2	3	4	5
-2	·	·	·	·	·	·	·	·
-1	·	·	·	·	·	·	·	·
0	·	·	·	·	1	·	·	1
1	·	·	·	·	1	2	2	·
2	·	·	·	2	3	1	·	·
3	·	1	1	1	·	·	·	·
4	·	·	·	·	·	·	·	·
5	·	·	·	·	·	·	·	·

HISTOGRAM OF RIGHT EDGE DIFFERENCES

COLUMN SHIFT

LINE SHIFT	-2	-1	0	1	2	3	4	5
-2	·	·	·	·	·	·	·	·
-1	·	·	·	·	·	·	·	·
0	·	·	·	·	·	·	1	·
1	·	·	·	1	·	2	·	1
2	·	·	1	·	4	·	1	·
3	·	1	·	2	·	1	·	·
4	·	·	1	·	·	·	·	·
5	·	·	·	·	·	·	·	·

Figure 4b. Left and Right Edge Differences Computed for Each Pair of Endpoints from Figure 4a (Histograms of these differences have peaks at (2, 2), the value of the translation of the object model.)

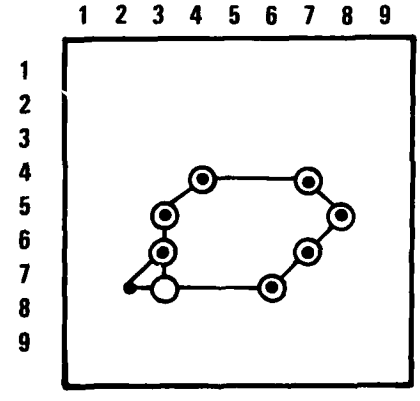


Figure 4c. Results of the Two-Dimensional Silhouette Matching (Note the alignment of the two object outlines after shifting the object model.)

As in the FSMA, separate histograms are computed for the left and right edges of the object. The edge which gives the greatest peak will be used to determine the correct translation. Since we consider all pairs of points from the object outlines, this method (unlike the FSMA) is independent of the starting positions of object outlines. The two-dimensional histogram algorithm requires only one iteration to find the best registration of the two outlines.

The noniterative technique has been applied to several frames from the data base. The results of matching two of the objects from the frames are shown in Figures 5 and 6. A partial view of each histogram is also shown in the figures. Note that the histograms for Figure 5 have peaks at different locations, indicating that only the right edge has been matched exactly. On the other hand, the peaks in the histograms for Figure 6 both fall at the same location, indicating both edges have been matched.

The results of the noniterative scheme compare favorably with those of the iterative scheme presented in the last report. Furthermore, the noniterative technique is computationally less complex than the iterative technique. When we compute the two-dimensional histogram for the noniterative scheme, we require N*M operations, assuming there are N lines in the first object and M lines in the second. (Note that operation in this sense may imply several arithmetic operations.) In order to compute the one-dimensional vertical displacement histogram used by the iterative scheme, it is also necessary to make N*M operations; since for each endpoint in the first object, we must see if any endpoint in the second object is in the same

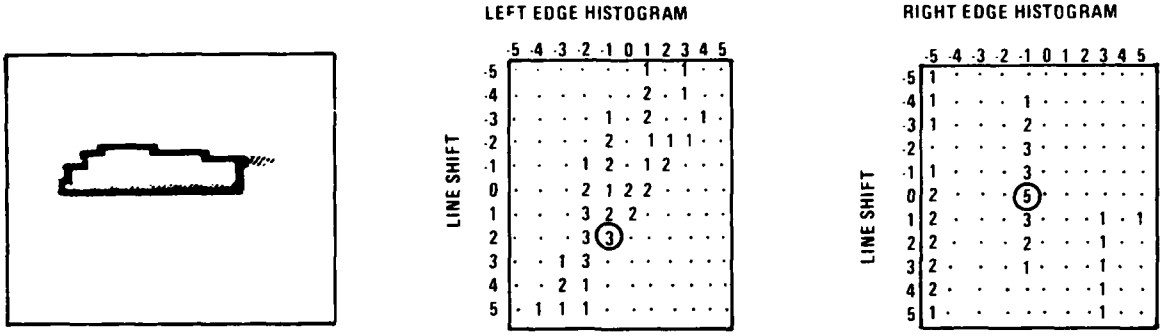


Figure 5. Two Object Outlines After Alignment (The proper alignment was determined by the peak in the right edge histogram.)

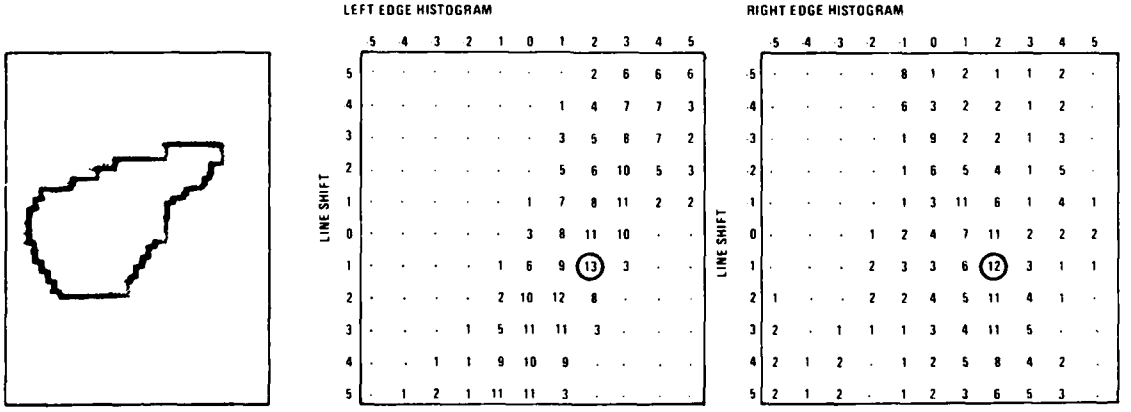


Figure 6. Two Object Outlines After Alignment (The proper alignment was determined by peaks in both the left and right edge histograms.)

column. If there is an endpoint in the same column, then the vertical displacement required to align those points is determined and added to the histogram. In addition to the vertical displacement histogram, we must compute a horizontal displacement histogram as well. This can be done in N operations. Therefore, the complexity of one iteration of the iterative scheme is $N * M + N = N * (M + 1)$ operations.

Since each iteration requires computing a vertical and horizontal displacement histogram and we may require as many as four iterations to find the precise alignment, the noniterative technique could reduce the computational load by as much as a factor of 4.

Both these algorithms, the FSMA and the two-dimensional silhouette-matching algorithm, are being implemented in the tracker simulation. The speed and accuracy of the two algorithms will be compared using the analysis program described in the following paragraphs, in order to find which algorithm satisfies the requirements outlined at the beginning of this section.

OBJECT-MATCHING ANALYSIS PROGRAM

In order to characterize the performance of the three object-matching schemes, an analysis program is being developed. The program uses the simple feature-matching algorithm and both the iterative and noniterative silhouette-matching algorithms to match two sets of objects. The objects in the two sets will be selected so that for each object in the first set there will be a matching object(s) in the second set. We will attempt to match each object

in the first set with all of the objects in the second set. By collecting intermediate results of the matching process, we will be able to determine how the algorithms perform when matching not only similar objects but also when attempting to match non-matching objects.

As previously mentioned, the analysis program will require two sets of objects. The objects in the first set will be chosen from several frames to present a variety of shapes, sizes, and intensity characteristics to the matching programs. The second set of objects will consist of all the objects from subsequent frames which have been determined to match those in the first set. The objects will be chosen so that all match types are represented one-to-one, one-to-many, many-to-one, and many-to-many.

Once these data sets have been created, the silhouette-matching algorithms will be applied to the two sets of objects. We will attempt to match each object in the first set with each object in the second set. For each iteration of the silhouette-matching algorithms, the following data will be collected:

1. The location of the peak of the difference histogram
2. The size of the peak
3. The size of the objects being matched
4. The size of the histogram
5. The shift determined by the algorithm

By comparing the expected sizes of the peaks in the difference histograms for the matching and non-matching objects, we will be able to set thresholds for these algorithms to provide accurate matches and low false match rates. We will also have empirical results concerning the convergence properties of the iterative algorithm.

The analysis program will also incorporate the simple feature-matching algorithm. This will allow testing of different feature sets and thresholds to determine the best feature-matching scheme. Results from the analysis program will be reported in subsequent progress reports.

SECTION 3

SCENE MODEL

The primary function of the scene model is to keep track of and infer information about objects in the scene as well as the platform dynamics derived from the analysis of the previous frames. More specifically, the scene model comprises:

- Platform dynamics (position and velocity)
- Object dynamics
- Object shapes and classifications
- Occlusion prediction
- Shape prediction
- Background prediction

The platform dynamics correspond to the motion of the sensor and the RPV (or the AAH) and its impact on the received image. Knowledge of the platform dynamics is useful both in finding the relative motion of targets with respect to the scene and in providing scene-track information to the platform gimbals if scene stabilization is required. Platform dynamics are computed from the positions of corresponding clutter (stationary) object matches.

Individual object positions computed by segmentation and object matching are used to compute the individual object dynamics over several frames. Object dynamics can be represented either relative to the sensor field of view or relative to the scene after the platform dynamics have been accounted for. The former is useful for multitarget tracking (say for laser designation) where only the positions of the target relative to the current field of view are desired. The latter also estimates the motion of the target relative to the scene (independent of the sensor motion) and permits target/clutter discrimination based on motion.

Because the scene model keeps track of all the object positions as well as the background characteristics in different regions of the image, it can be used to predict the occlusion of objects that are moving toward each other, an object which moves into a low-contrast background, etc. The shapes of occluded objects can also be predicted so that the object matcher can use the predicted shapes to perform better matches in successive frames. Furthermore, the artificial intelligence capability of the scene model will allow inference of the target shape from its segmentations in previous frames. For example, if multiple segments of an object appear to move together over several frames, then the inference is that they belong to the same object.

During this reporting period, a data structure for representing the segmented objects and the relationships between the objects was developed. Within this data structure, the problems of occlusion and missegmentation can be represented so that the object matcher can efficiently deal with them. The data structure also allows for new interobject relationships to be defined and incorporated into the tracker.

SCENE MODEL DATA STRUCTURE

The data structure used to represent the objects will be used to satisfy the requirements listed in the introduction to this section. It must be able to represent all the relationships between objects. Furthermore, since the scene model development is an evolving process, the model must be able to adapt as more capabilities are required from it.

The data structure represents object models and segmented objects as nodes in a graph. Each node contains fields which describe the object and its relationship to the other objects in the scene. The relationships can be viewed as edges in the graph.

The general data structure is illustrated in Figure 7. Extracted objects from a given frame are represented by the circles, and the relationships between the objects are represented by the arrows. Note that objects can be linked to objects in the same frame or to objects in subsequent frames. Interframe links are determined by the matching algorithms. Intraframe links are used to point to different components of the same object, such as the tread, motor, and barrel of a tank or the several components of an extended background region.

The application of this data structure to the matching problem is straightforward. If object A in frame 1 matches object B in frame 2, we will set up a link between A and B as shown in Figure 8. Similarly, many-to-one and one-to-many matches can be represented by several

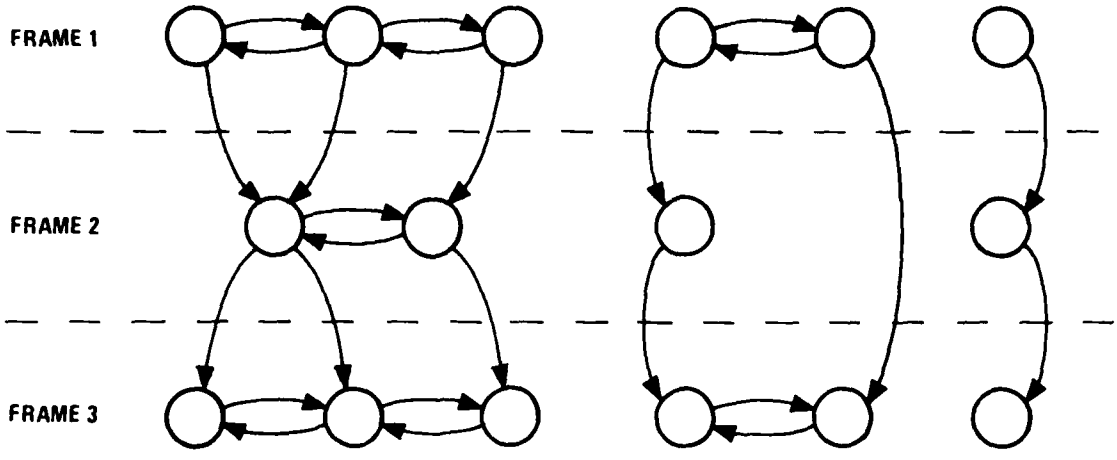


Figure 7. Example of the Scene Model Data Structure Showing Interframe Object Matches and Intraframe Object Relations

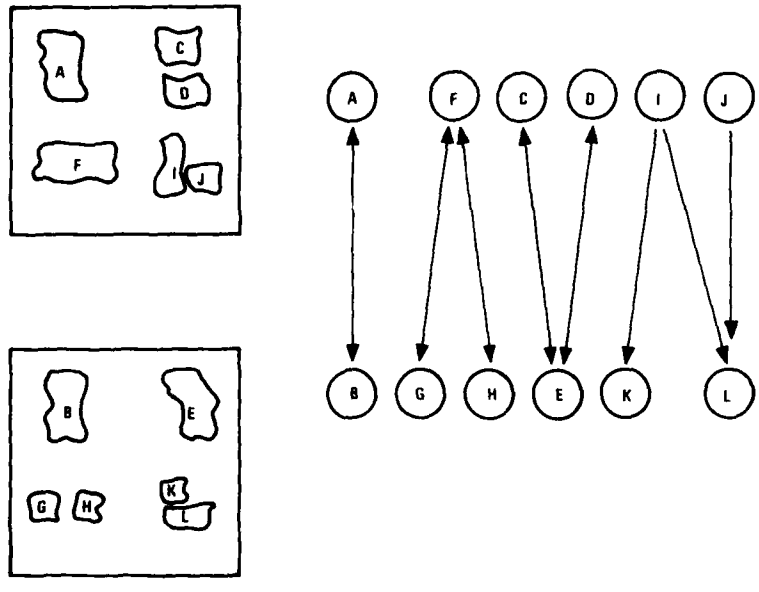


Figure 8. Representation of One-to-One, One-to-Many, Many-to-One, and Many-to-Many Matches with the Scene Model Data Structure

links as shown in Figure 8. These links can be used to find all the different ways this object has been segmented over several frames. This gives us a better chance of finding a matching object in the current frame by not only allowing matching between the current frame and the last frame but also between the current frame and several previous frames.

The problem of occlusion can also be represented in this structure; it is a special case of the many-to-one match. This is illustrated in Figure 9. The object C will be identified as an object composed of two object models, that is, the result of A moving as to occlude B. Therefore, matching in the next frame will not be done using object C, since that object will have changed shape due to the further motion of A. However, matching will be done using the unobscured edges of A and B, which are the components of object C. In this manner occlusion can be recognized and resolved by the matching function.

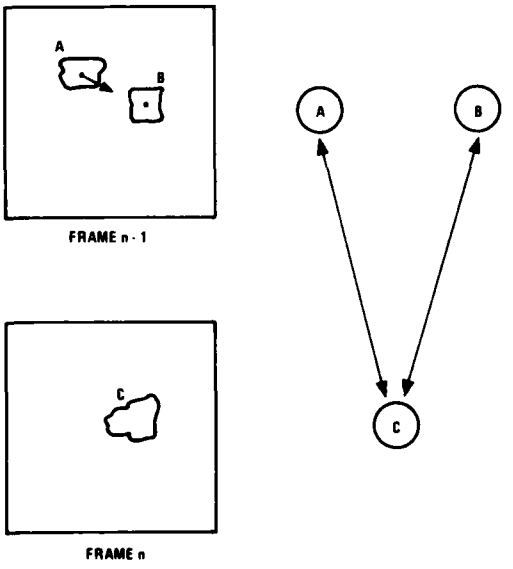


Figure 9. Scene Model Data Structure Representation of Occlusion

Furthermore, the graph structure is very flexible. The nodes can be allowed to grow as required and to accommodate additional interobject relationships which will be implemented.

The implementation of this data structure in the system software is discussed in the next section.

SECTION 4

SYSTEM SIMULATION

The techniques which have been developed during this reporting period have been incorporated into a complete system simulation of the advanced target tracker system in the Honeywell Image Processing Facility. This simulation allows the evaluation of the algorithms as they are developed in the system context. This system simulation will be expanded as new algorithms and software are developed for such factors as occlusion prediction, target/background signature prediction, and advanced scene models.

A block diagram of the current system simulation is shown in Figure 10. The simulation currently consists of the following software modules:

- PATS segmentation
- Simple object-matching
- Fast silhouette-matching

In the system simulation, the PATS segmentation is applied to the two input frames. This produces a list of object outlines and features which will be matched. The object-matching algorithms match objects between the two frames to find the interframe scene motion. The silhouette-matching algorithms match all the objects which are present in both scenes to find their exact displacement. The results of scene motion model and object matching are then combined to yield an estimate of the interframe object motion.

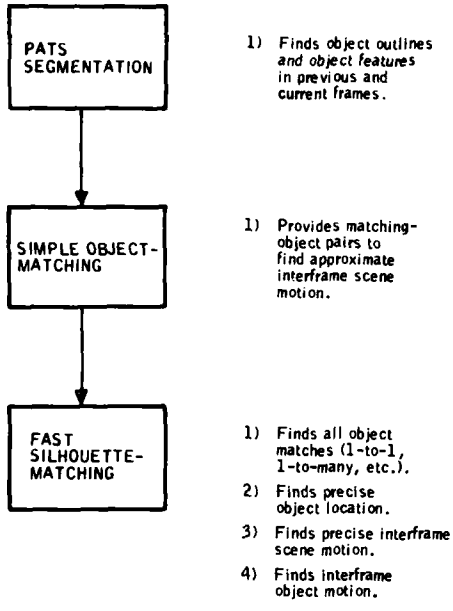


Figure 10. System Simulation Block Diagram

During this reporting period, the scene model data structure described in the previous section was incorporated into the system simulation. In order to efficiently implement the data structure, a dynamic memory allocation scheme was designed. It allows a large portion of memory to be divided into small blocks and allocated to the simulation. The small blocks are used to represent the objects and links within the data structure. This section discusses the implementation of the scene model data structure and dynamic memory allocation scheme.

SCENE MODEL IMPLEMENTATION

As mentioned in the previous section, the object models and segmented objects are represented as nodes in a graph and relationships between objects as edges in the graph. Each node in the graph consists of a number of fields. These fields contain information concerning the object such as area, contrast, location, and shape. The edges of the graph are represented as pointer fields within the nodes. They point to the node on the other end of the edge.

In general each object (node) will have a different number of edges entering it. Furthermore the number of edges entering a node may change as more information about an object is gathered. Therefore, there is no fixed size for an object node. The size of the nodes must be allowed to increase as required. In addition, some nodes may become inactive as object models are updated and replaced. This implies that we must have the capability of reusing inactive nodes, or we would soon exhaust our node storage.

The previously mentioned requirements (variable-sized nodes and reuse of inactive nodes) can be implemented using a dynamic memory allocation scheme. These algorithms create a structure within a block of memory which allows the user to satisfy these requirements. The algorithms we have implemented are modified versions of those found in reference 3.

---

[3] Ellis Horowitz and Sartaj Sahni, "Fundamentals of Data Structures," Computer Science Press, Inc., Woodland Hills, California, 1976.

Those algorithms provide for the reuse of inactive nodes. We have added several routines to allow variable-sized nodes to grow as the simulation progresses. The algorithms are currently implemented in FORTRAN in the tracker simulation. However, these algorithms are independent of a particular programming language or computing hardware. Therefore, they would be suitable for implementation in any processor. The following paragraphs describe the dynamic memory allocation algorithms.

DYNAMIC MEMORY ALLOCATION

The dynamic memory manager is a set of utilities which allocate blocks of memory to the requesting routines, read and write locations within the memory block, and return blocks of memory to a pool of available memory. These routines also have the capability to allow the allocated blocks of memory to grow as a routine requires more storage. This allows efficient use of a limited amount of main memory.

The memory manager maintains a linked list of all the blocks of memory that are currently unused as shown in Figure 11. An expanded view of one of the unused blocks is shown in Figure 11b. The first and last words contain the size of the block of memory. The second and third words contain pointers to the next and previous entries in the list. The forward and backward links, stored in the second and third words, are required in order to add and delete memory from the available space list. When a request for memory is received, this list is searched until a block is found that can fill the request. The requested memory is given to the user from that block. If a sufficient portion of that block is left after the allocation, it remains in the list of available memory.

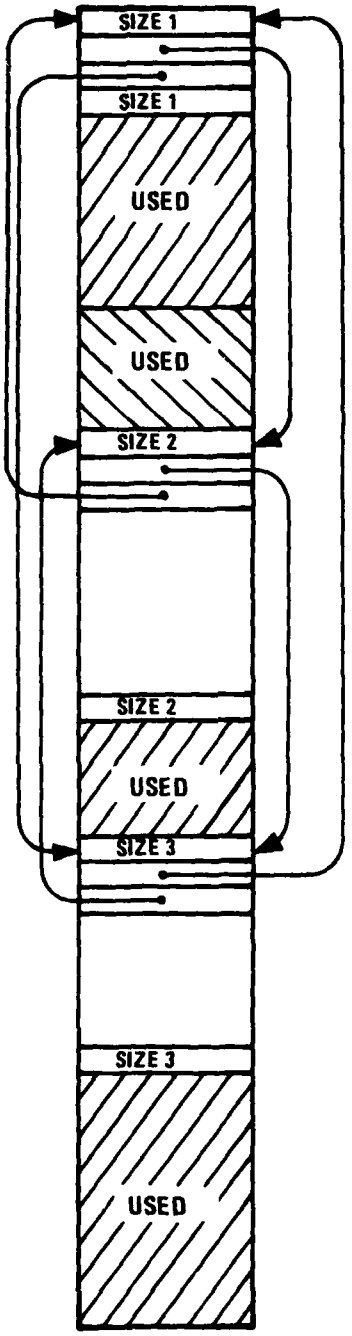


Figure 11a. Structure of the Available Blocks of Memory within the Memory Pool (The available blocks are organized into a doubly linked list.)

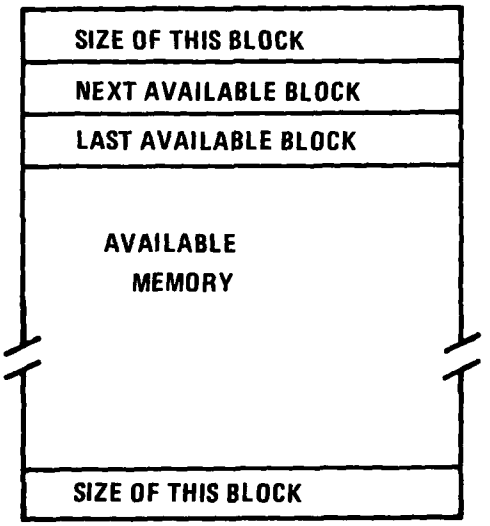


Figure 11b. Expanded View of an Available Block of Memory Showing the Links and Fields Used by the Dynamic Memory Manager

The user is able to access the allocated memory by retaining a pointer which identifies the allocated block as shown in Figure 12. The user program pointers are indices into a user connection table which gives the actual memory location of the block. The user connection table is required so that the memory manager can change the location of a given block of memory without informing the user. When a block is moved, the memory manager simply updates the connection table using the link stored in the second word of the used block. Then the next time the user references that block, the new memory location can be found. Since the number of pointers in the user program is variable, the user connection table must be of variable size. For this reason the connection table itself is stored within the memory pool.

When a user is finished with a block of memory, it can be returned to the available memory pool. A check is made to see if any of the adjacent memory is also free; if so, the returned block is merged with all adjacent free blocks to form a larger free area. The new free area is then added to the list of unused memory blocks.

If the memory manager receives a request for memory which is larger than any of the available blocks, it attempts to create a large enough block through a process known as garbage collection. This process moves all the used blocks of memory to the lower portions of the memory pool. This creates one large available block of memory at the upper end of the memory pool. If this block cannot satisfy the request, then it cannot be honored until the user returns more of the blocks.

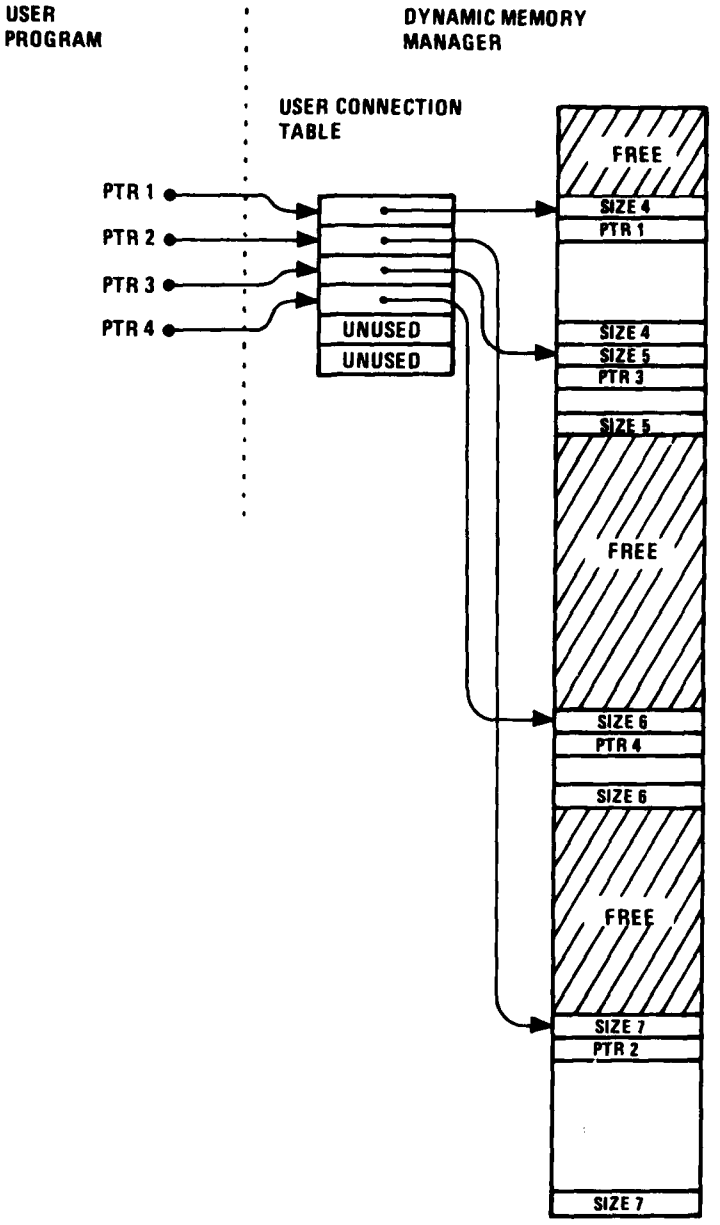


Figure 12. Structure of the Used Portions of the Memory Pool (The user references data stored in the pool by retaining a pointer into the user connection table. The entry in this table allows the memory manager to locate the requested block of memory.)

The memory manager can expand an allocated block of memory at the request of the user in two ways: 1) by using an adjacent free block of memory, or 2) if one is not available, by transferring the block to a different, and larger, unused block of memory.

SECTION 5

PATS SIMULATION

The baseline segmentation approach for the advanced target tracker simulation is the Honeywell PATS. The PATS simulation duplicates the detection and recognition phases of the PATS hardware. As part of the Advanced Target Tracker Concepts contract, this simulation was transferred to an IBM 360 at NV&EOL. The conversion of the program to use NV&EOL's EIKON image file system was completed at Honeywell's Image Processing Facility. The final program modifications and program checkout took place at NV&EOL during this reporting period.

Also during this period the PATS algorithms have been modified to prevent the breakup of extended targets. Previously, the PATS simulation and hardware would break large objects into several smaller segments. Often this breakup would not be consistent between frames. This made interframe matching very difficult. The algorithms were modified to combine objects which had been incorrectly broken up.

PATS SIMULATION TRANSFER

Appendix A describes the PATS simulation routines and provides operating instructions.

CHANGES TO THE PATS SIMULATION

As described in the previous section, PATS generates object intervals based on the bright, cold, and edge signals. These intervals are then combined on a line-by-line basis to form object bins. If no intervals are added to a bin after a given number of lines, then the bin is closed. This strategy will break up large, irregularly shaped objects such as the one in Figure 13a.

The object intervals for that object are shown in Figure 13b. As these are processed line by line, two bins will be generated, one for each hump in the object. When the interval in the fourth line is processed, it will not match any of the existing bins due to the great difference in width between the interval and the bins. This is also true for the intervals in lines five and six. The resulting objects are shown in Figure 13c. The original object has been broken into three smaller objects. The smaller objects are more likely to be misclassified as targets than the large object shown in Figure 13d which has been correctly segmented.

The breakup of objects, such as the one in Figure 13, obviously distorts the shape of the silhouette. This causes the silhouette matching programs to either fail to find a match or to incorrectly align the objects and generate false moving objects.

The solution to this problem is to verify, before closing a bin, that there are no adjacent bins with similar characteristics. For example, before closing bin no. 1 in Figure 13c, we would search for adjacent bins. Bin 3 would satisfy the requirements, and bins 1 and 3 would be combined. Similarly, before closing bin 2 we would search for adjacent bins, and bins 2 and 3

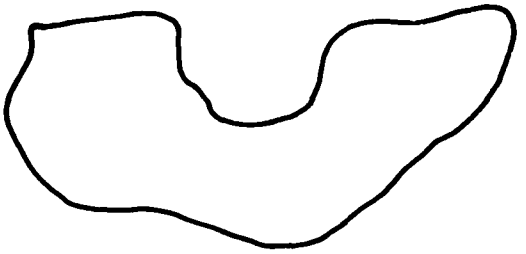


Figure 13a. Original Object Outline Presented to PATS Simulation

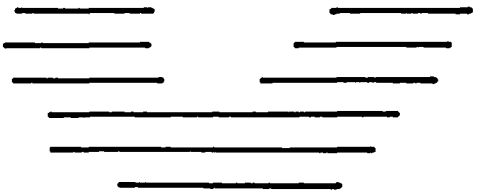


Figure 13b. Object Intervals Found by the Simulation

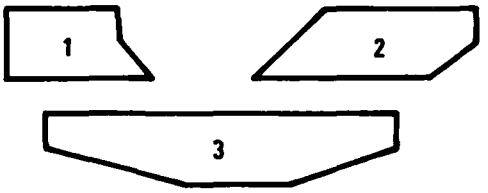


Figure 13c. Segmented Object Outlines Found by the Simulation (Note the breakup of the original object.)

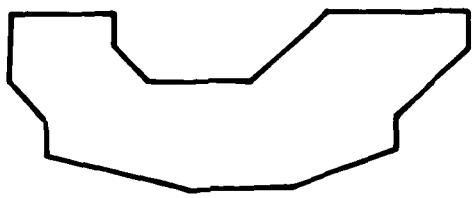


Figure 13d. Correct Segmentation of Original Object

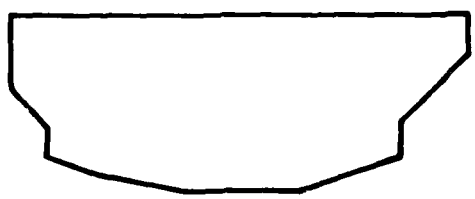


Figure 13e. Segmentation Found by the PATS Simulation After Modifications to Prevent Large Object Breakup

would be combined. Due to the nature of the form in which objects are represented in PATS (that is, as start and end points of unbroken horizontal intervals), the concavity of the object in Figure 13a cannot be represented. Therefore, the result of the PATS segmentation will be as shown in Figure 13e.

The results of the simulations, before and after modification, are shown in Figures 14, 15, and 16. Figure 14 is the FLIR frame. Figure 15 shows the PATS segmentation before the modifications to prevent the object breakup. Figure 16 shows the segmentation after modification. The affect of the modifications is clearly seen in the outlined regions of the segmented frames.

Figure 14. FLIR Image Used to Illustrate PATS Segmentation Improvements

Figure 15. Current PATS Segmentation Simulation for Frame from Figure 13. (Note the breakup of the groups of trees in the foreground.)

Figure 16. PATS Simulation Results After Modifications to Prevent Object Breakup (Note the larger objects compared to Figure 14. The dotted portions of the objects represent those parts of the larger objects which could not be stored by the simulation due to a 70-line limit on the object size.)

SECTION 6

PLANS FOR FUTURE REPORTING PERIODS

This section outlines the program plans for future reporting periods. Particular emphasis will be placed on:

- Scene model
- Target recognition and homing techniques
- Data base expansion

SCENE MODEL

The scene model which now contains a platform displacement estimator and a flexible data structure for representing object relationships will be extended to include object occlusion prediction and resolution. Also, Kalman filtering techniques will be employed to give robust estimators of scene and object motions. Multiple component association rules will be derived.

TARGET RECOGNITION AND HOMING TECHNIQUES

The baseline PATS segmentation simulation has been changed to give better segmentation results for input to the classification algorithms. In future periods syntactic techniques will be explored for close-in objects, using multi-components which are obtained by sequential refinement of single frame segmentations.

DATA BASE EXPANSION

The data base will be continually expanded to include challenging examples of occlusion and varied background signatures.

APPENDIX A

PATS SIMULATION ROUTINES AND OPERATING INSTRUCTIONS

APPENSIX A

PATS SIMULATION ROUTINES AND OPERATING INSTRUCTIONS

The PATS is an automatic target screener that can operate with first generation thermal imagers employing common module components. PATS will reduce the task loading on the thermal imager operator by detecting and recognizing a limited set of high priority targets at ranges comparable to or greater than those for an unassisted observer. A second objective is to provide enhancement of the video presentation to the operator.

The PATS simulation is a set of FORTRAN routines which simulate the detection and recognition phases of the PATS hardware. The simulation consists of a main program which calls one of the four subprograms which perform the segmentation and classification of an input EIKON[4] image. A block diagram of the simulation appears in Figure A-1. A description of each of the four subprograms follows.

SUBPROGRAM DESCRIPTIONS

PATS1

The input to this routine is an EIKON image. The program uses a two-dimensional recursive background estimator to find the expected value of

---

[4] "Introduction to EIKON, Night Vision Laboratory Image Processing System," Research and Development Technical Report, ECOM-7058, United States Army Electronics Command, Fort Monmouth, New Jersey, November 1976.

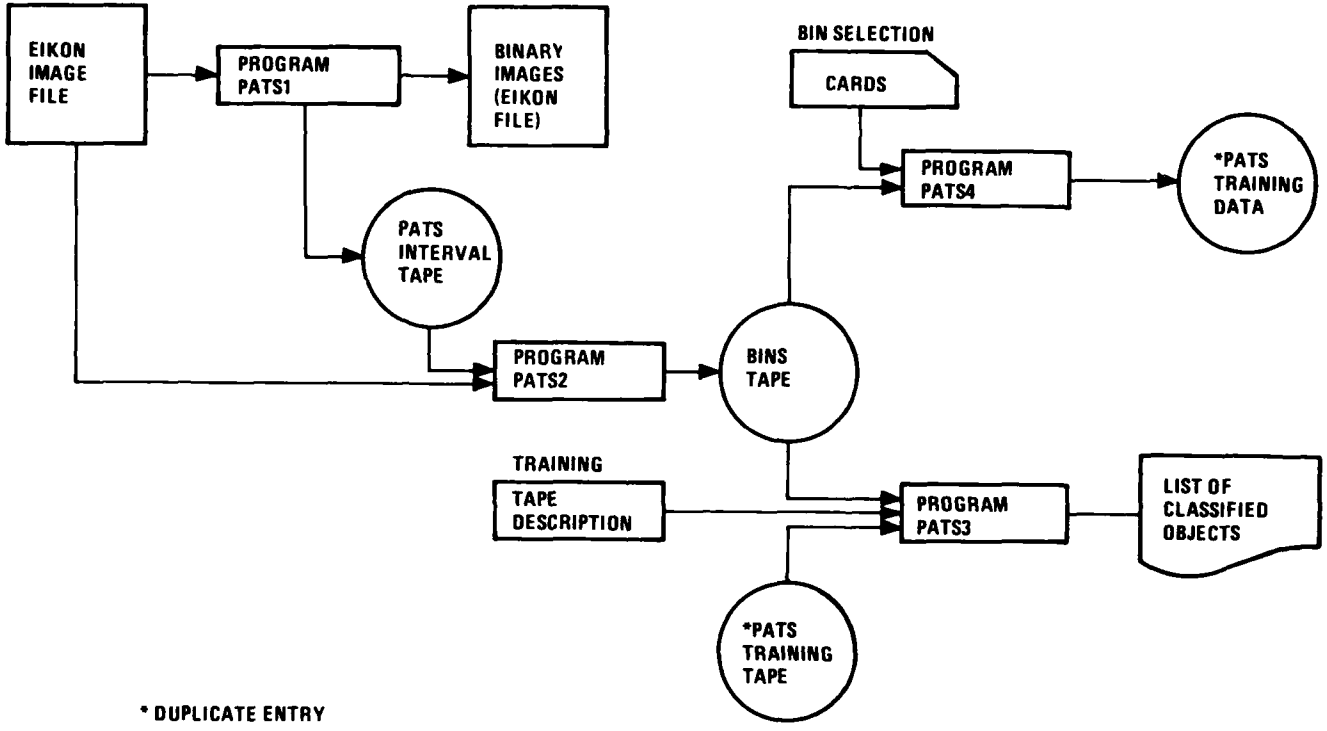


Figure A-1. Block Diagram of PATS Simulation Showing Required Data Items and Outputs for Each Subprogram

the background at a given pixel. Those pixels which are significantly hotter (brights) or colder than the background are found. A vertical edge operator similarly finds those pixels which are located near significant edges. The bright, cold, and edge pictures are scanned from left to right and right to left in order to find object intervals. An interval begins at a pixel which shows both significant edges and either brights or colds. The interval is turned off when there are no longer bright or cold pixels.

The output of PATS1 is an EIKON image containing four-bit pixels corresponding to the binary edge, bright, cold, and interval images which can be displayed. Also, a file of the line number, column number, width, and other features for each interval is created.

PATS2

The inputs to this routine are the file of interval features produced by PATS1 and the original EIKON image.

The program reads intervals from the interval tape and associates similar intervals. A collection of similar intervals is known as a bin. If no intervals are added to a bin after a certain number of lines, the bin is closed and bin features are calculated. These include the outline, moments, Fourier descriptors, etc. These features are then output to the BINS tape.

PATS3

The inputs to this routine are the BINS tape created by PATS2 and a PATS training tape created by PATS4. The program also needs a set of four cards describing the training tape.

The program first reads the cards and training tape and then stores the selected object features and object classes. Then the BINS tape is read. Each bin is first converted to the format used by the classification subprograms. Then it is passed through the clutter rejection classifier. If the object passes this test, it will be classified as a target; if not, it will be clutter. Those objects which pass the clutter rejection classifier are run through the K-nearest neighbor (KNN) classifier to determine the target class, either tank or APC. The KNN classifier finds the K-nearest neighbors from the training tape in the selected feature space. The target class is determined by polling the K-nearest neighbors. The results of the classification is displayed at the line printer.

PATS4

This routine operates on a BINS tape created by PATS2 and a set of cards to generate a training tape. Each card consists of a bin number and target class. As a card is read, the bins tape is searched (only in a forward direction) until that bin is found. The bin is converted to the classifier format and then written to the training tape. The output then is a tape of classified objects and features to be used for input to PATS3.

DESCRIPTION OF DATA FILES

EIKON Image File

This is the frame to be segmented and classified. The whole frame or any subframe may be processed and is controlled by the EIKON INPUT control card.

Binary Image File

This is one of the outputs of the PATS1. It is an EIKON image, and its size must be the same as the subframe processed by PATS1. That is, if a 100 x 100 subframe of a 512 x 512 image is processed by PATS1, the output image size determined by the EIKON OUTPUT control card must be 100 x 100. The four binary images are coded into a four-bit pixel as follows:

Bit	Contents
1	Edge
2	Bright
3	Cold
4	Interval

The word length of the EIKON image is 16 bits even though only the least significant four bits are used.

Interval Tape

This output of PATS1 contains one record for each interval found by the PATS1 program. Each record is written in (1X,10I4) format. The 10 fields are defined as follows:

1. Absolute line number of interval
2. Absolute starting column number
3. Width of the interval
4. Number of edges in expanded interval where the expanded interval extends 2 pixels in both directions over the given interval

5. Number of brights (- [minus] for colds) in the interval
6. Background value at left endpoint
7. Average interval intensity
8. Peak interval intensity
9. Minimum interval intensity
10. Edge at Start; Edge at End
    This word indicates if a significant edge occurs at the start, end, or both of an interval as follows:

Bit	Contents
1	Edge at Start
2	Edge at Middle
3	Edge at End

The last record in the file contains 999 in all data fields. This is followed by an end-of-file mark.

BINS Tape

This output of PATS2 contains one record for each bin (object) found by the program. The record is written as a 300-word real array and a 74-word real array (that is, WRITE (10) (DAT(K), K = 1,300), (RDA(K), K = 1,74). The contents of the array are given in a later section titled BINS TAPE DATA. The last record in the file contains 999 in the first 100 words. This is followed by an end-of-file mark.

Bin Selection Cards

The purpose of these cards is to select certain bins from a given bin file and to transfer these to a training tape for use by the nearest neighbor classifier. The cards are inserted after the GO card for PATS4.

Each card is in 2I4 format. The first field is the bin number, and the second field is the bin type.

Only those bins selected by the cards will be converted to the classifier format. The given bin type is included in the classifier format. Note: the number of a bin is implicitly defined by its position in the file with the first bin of the file being bin no. 1, second being bin no. 2, etc.

PATS Training Tape

This is a file of bins, in classifier format, which have been manually identified as targets. These objects and features are written using an unformatted write of a 300-word integer array. The contents of the 300-word array are given in a separate subsection titled Classifier Format.

Training Tape Descriptor Cards

This input to PATS3 consists of exactly four cards. All the cards are written in 20I4, although not all 20 fields are used. The first card is a type-to-class conversion card. This card maps bin types stored on the training tape to target classes. This allows the user to try a variety

of classification schemes. For example, he can try to classify tanks, APCs, and trucks separately, or perhaps tanks and APCs as one class and trucks as another. The current training tape only contains two classes of targets. The first field of the type-to-class conversion card contains the number of classes (NCL), and the second entry contains the number of types (NT). Following the second entry are NT more entries. Each of these NT values are in the range [0, NCL]. The first of these is the class corresponding to type 1, the second to type 2, etc. If a certain type has no class, then a 0 is entered.

The second card is the sample description card. The first entry is the number of samples (NS) on the training tape. The training tape will be read until NS samples are read or until an end of file is encountered. The second entry is the number of features (NF) to be taken for each object. This represents the maximum dimension of the feature space. The next NF entries are the numbers of the features to be used. The feature numbers are listed in the description of the classifier format.

The third card is a feature subset selection card. This card selects a subset of the features indicated on the second card to use in the KNN classifier. The first entry is the number of features (NSS) which must be less than NF. Following the first entry are NSS entries which point to features listed on the previous card. The fourth card specifies the number of neighbors to use in the classification (KNEI). This number must be less than NS, the total number of training samples.

The deck in Figure A-2 specifies two target classes from nine bin types. Types 1, 5, and 9 correspond to Target Class 1. Types 2 and 6 correspond to Target Class 2. Types 3, 4, 7, and 8 do not correspond to either target class. The second card specifies that the size of the training data base is 227. Six features will be retrieved from the tape those corresponding to features 179 to 184. However, only four will be used in the classification, the first, second, fourth, and sixth, or 179, 180, 182, and 184. The fourth card specifies that the eight nearest neighbors will be polled to determine the class of a given object. This sample is the recommended set to use when the training tape supplied by Honeywell is used.

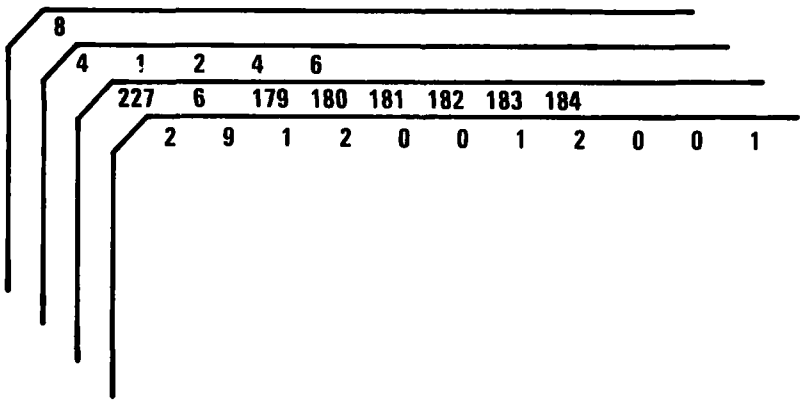


Figure A-2. Example of Training Tape Descriptor Cards

List of Classified Objects

This is the output from the PATS3 program. Two lines will be output to the line printer for each object in the input bin file. The first line gives the result of the clutter rejection classifier in the following format:

OBJECT NO. n CLUTTER REJECTION RESULTS q

where n is the object (bin) number and q is the classification results

1: target

2: clutter

The second line gives the result of the target classification in the following format:

OBJECT NO. n TARGET CLASSIFICATION RESULTS r

where n is the object number and r is the classification results as determined by the training tape supplied and deck shown in Figure A-2

1: tank

2: APC

3: Unused

4: Clutter

Note that the object number is implicitly defined by that object's position within the bin file. Also, these target classes can be modified by using different training sets containing different target types or by using a different type-to-class conversion card in the training tape descriptor deck.

RUNNING THE SIMULATION

System Job Control Cards

ASSGN cards are needed to associate the FORTRAN reads and writes with the correct data set. These are described as follows:

// ASSGN SYS005, SYSLST--This causes writes to unit 8 to appear at the line printer; Units 8 and 6 are used by the simulation to produce hard copy output. This card is needed for all PATS simulation runs.

// ASSGN SYS006, TAAAAA,T--This assigns the FORTRAN unit 9 to tape AAAAA. This should be the PATS interval tape. This card is needed when running PATS1 or PATS2.

// ASSGN SYS007, TBBBBB,T--This card assigns FORTRAN unit 10 to tape BBBBB. This tape should be the BINS tape. This card is required when running PATS2, PATS3, or PATS4.

// ASSGN SYS008, TCCCCC,T--This card assigns FORTRAN unit 11 to tape CCCCC. This tape should be the PATS training tape. The card is required when running PATS3 or PATS4.

// ASSGN SYS011, TDDDDD,T--This card assigns tape DDDDD to EIKON unit 1. This tape should be the input image; although if the user wishes, the INPUT control card can specify input/output from any units without altering program performance.

// ASSGN SYS012, TEEEEE, T--This card assigns tape EEEEE to EIKON unit 2. This tape should be the binary output image; although if the user wishes, the OUTPUT control card can specify output to any of the other EIKON units without altering program performances.

EIKON JOB CONTROL CARDS

When running PATS1, both INPUT and OUTPUT control cards are required. The LOC field on these cards must agree with the ASSGN job control cards. That is, if the input tape is assigned to SYS011, then the INPUT control card should have a 1 in the LOC field. The extent specifications on the INPUT card can be used to process the whole frame or any subframe included within the frame. The increment fields IDX and IDY must be one for proper program execution.

The OUTPUT control card defines the binary images EIKON file. The size of this file must be identical to the size of the frame to be processed; that is, IXC = JXC and IXC = JYC.

All four of the PATS routines are loaded in a single phase when the phase name card PATS is processed. In order to run one of four routines, the 10th field of a GO card should contain a number in the range [1, 4] which identifies the particular routine to run. The other fields of the GO cards are explained in the following paragraphs.

GO Card Fields for PATS1

As previously explained, the 10th field must contain a 1. The seventh to ninth fields are not used by the PATS1 program. The first six fields are used to control six logical variables as follows:

SS1 = IGO(1).EQ.1

SS2 = IGO(2).EQ.1

SS3 = IGO(3).EQ.1

SS4 = IGO(4).EQ.1

SS5 = IGO(5).EQ.1

SS6 = IGO(6).EQ.1

If SS1 is true, then each time an interval is added to the interval tape the 10 interval features are also printed on the line printer. Each feature is preceded by a four-character label describing the field. If SS1 is false, this line printer output is suppressed.

If SS2 is true, then the program will use a set of default filter constants and thresholds. The parameters and default values are listed below:

Variable Name	Function	Default Value
ALP, BET	Background filter constants	(0.9)
K	Lines and columns required to initialize background filter	(20)
C	Hot/cold threshold	(1.5)

Variable Name	Function	Default Value
A	Upper hot/cold threshold	(1000.0)
EC	Edge threshold	(1.5)
EA	Upper edge threshold	(100.0)
W,H	Width and height of edge operator	(3,1)
SON	Edge threshold for interval turn on	(3)
TON	Bright threshold for interval turn on	(2)
SOFF	Size of window for turn off	(4)
TOFF	Threshold for turn off	(3)
ALFA	Edge filter constant	(0.9)

If SS2 is false, the program expects a data card immediately following the GO card. This card is punched in 14I4 format and contains 14 integer values, I1 to I14, which will be multiplied by an appropriate scaling factor and used by PATS1. (Note that these values become the new default parameters if PATS1 is executed again without reloading the phase PATS.) The filter constants and thresholds are determined as follows:

$$ALP = I1/100.0$$

$$BET = I2/100.0$$

$$K = I3$$

$$C = I4/100.0$$

A = I5/10.0

EC = I6/100.0

EA = I7/10.0

W = I8

H = I9

SON = I10

TON = I11

SOFF = I12

TOFF = I13

ALFA = I14/100.0

If SS3 is true, then for each line of the input image the following line printer output appears:

1. The line number relative to the first line processed
2. The value of the background estimator for each pixel in the current line
3. The intensity of each pixel in the current line
4. The bright signal for each pixel from the current line
5. The cold signal for each pixel from the current line
6. The edge signal for each pixel from the last line
7. The interval signal for each pixel from the last line

If SS3 is false, this output is suppressed. (Note that when SS3 is true there is considerable line printer output.)

If SS5 is true, then for each line of the input image the following line printer output appears:

1. A header containing the absolute line number of the current line
2. The forward (left to right) and backward (right to left) interval signal; for each pixel in the input line, the value of the interval signal at that point is printed:

   0--no interval

   1--hot interval

   -1--cold interval

Note that the backward interval signal is displayed in reverse order; that is, the first pixel printed corresponds to the last pixel in the picture. If SS5 is false, this output is suppressed.

If SS6 is true then after every 10 lines the following output appears at the line printer:

1. The average absolute difference of the input intensities and the background estimator computed after processing each line
2. The standard deviation of the edge operator in each of the last 10 lines
3. The filtered second moment of the edge operator in each of the last 10 lines
4. The value of the edge threshold for each of the last 10 lines

If SS6 is false, then this output is suppressed.

GO Card Fields for PATS2

Only the EIKON INPUT control card is needed when running PATS2. The parameters contained on the INPUT card should be the same as on the INPUT card for PATS1.

The GO card parameters for PATS2 are as follows:

IGO(1) Number of missing intervals in order to close a bin

IGO(2) Edge discontinuity threshold

IGO(3) Minimum length of a bin

IGO(4) Minimum length of an interval

IGO(5) Maximum length of an interval (not used)

IGO(6) Maximum difference in interval widths

IGO(7) Display flag:

0 = no line printer output

1 = line printer output of object outlines and calculated features

IGO(8-9) Unused

IGO(10) Must be set equal to 2 in order to run PATS2

Since there is no default option in PATS2, each parameter must be specified on the GO card. The following is a list of recommended values:

1. Number of missing intervals 4, 5, or 6
2. Discontinuity threshold 3
3. Minimum bin length 4, 5, or 6
4. Minimum interval length 3, 4, 5, or 6
   Note: larger values for items 3 and 4 result in fewer extracted objects.
5. Not used in the program
6. Maximum difference in interval widths 18, 19, 20, 21, or 22
7. Display flag 0 or 1

The line printer output from PATS2 (when the seventh GO parameter is a 1) consists of a portion of the original frame around the target or the message BAD RETURN FROM MMCALC. When the message appears, the extracted object was too large to calculate moments. In that case, all moment features will be set to zero. When the subframe appears, the column and row of the upper left corner of the subframe will be output also. Following the subframe will be a list of features and values. A full description of these features is given in the description of the BINS tape. If the BINS ARE FULL message appears at the line printer, it signals that all 50 bins are active and the input interval does not match any of the active bins. Normally this would result in opening a new bin. However, if all bins are active, the interval is simply discarded and the next interval is processed.

GO Card Fields for PATS3

Since there are no EIKON input or output files associated with PATS3, no EIKON control cards are necessary except those which load the PATS phase and the GO card to execute PATS3.

As previously described, the tenth field of the GO card to execute PATS3 must be a 3. The other fields are used as follows:

1 to 3	Unused
4	IGO(4) controls a logical variable as follows: SS4 = IGO(4).EQ.1
5 to 9	Unused
10	Must contain a 3

If SS4 is true, then the KNN classifier uses a City Block distance calculation. If SS4 is false, the Euclidean distance is used. The recommended value for IGO(4) is ∅ (zero).

GO Card Fields for PATS4

Since there are no EIKON input or output files associated with PATS4, no EIKON control cards are necessary except those which load the PATS phase and the GO card to execute PATS4.

The only GO card parameter needed to execute PATS4 is in the tenth field. This field must contain a 4.

Sample Deck to Run PATS1

```
// JOB  (Accounting Data)
// ASSGN  SYS005, SYSLST
// ASSGN  SYS006, TAAAAA, T        (Insert correct interval tape
                                    number)
// ASSGN  SYS011, TDDDDD, T        (Insert correct input tape
                                    number)
// ASSGN  SYS012, TEEEEE, T        (Insert correct output tape
                                    number)
// EXEC  EIKON
   REWIND  1                       (Insert correct EIKON tape
   REWIND  2                        positioning commands)
   PATS
   GO      0 1 0 1 0 0 0 0 0 1
   REWIND  1
   REWIND  2
   END
/*
/&
```

Sample Deck to Run PATS2

```
// JOB     (Accounting Data)
// ASSGN   SYS005, SYSLST
// ASSGN   SYS006, TAAAAA,T          (Insert correct interval
                                      tape number)
// ASSGN   SYS007, TBBBBB,T          (Insert correct BINS
                                      tape number)
// ASSGN   SYS011, TDDDDD,T          (Insert correct input
                                      tape number)
// EXEC   EIKON
   REWIND   1                        (Insert correct EIKON
                                      tape positioning com-
                                      mands)
   PATS
   GO       6  3  6  4  30  18  1  0  0  2
   REWIND   1
   END
/*
/&
```

Sample Deck to Run PATS3

```
// JOB     (Accounting Data)
// ASSGN  SYS005, SYSLST
// ASSGN  SYS007, TBBBBB, T          (Insert correct BINS tape
                                      number)
// ASSGN  SYS008, TCCCCC, T          (Insert correct training
                                      tape number)
// EXEC   EIKON
   PATS
   GO        0  0  0  0  0  0  0  0  0  3
    2  9  1  2  0  0  1  2  0  0  1
   227  6  179  180  181  182  183  184
     4  1  2    4    6
     8
    END
/*
/&
```

Sample Deck to Run PATS4

```
// JOB     (Accounting Data)
// ASSGN   SYS005, SYSLST
// ASSGN   SYS007, TBBBBB, T          (Insert correct BINS tape
                                       number)
// ASSGN   SYS008, TCCCCC, T          (Insert correct training
                                       tape number)
// EXEC    EIKON
   PATS
   GO        0  0  0  0  0  0  0  0  0  4
    3   1
    7   2
   15   1
   21   1
   -1
   END
/*
/&
```

BINS TAPE DATA

As previously mentioned, the BINS tape consists of one record per bin. This record is written as a 300-word real array followed by a 74-word real array.

Part 1, 300-Word Real Array

1. TOTAL LENGTH

$$L_n - L_1 + 1$$

Where:

L_n = Last line number

L_1 = First line number

2. ACTIVE LENGTH

The number of active intervals

3. ACTIVE AREA

$$\sum_{i=1,m} W_i$$

Where:

W_i = Width of ith active interval

m = Active length

4. CENTER COLUMN

$$\frac{1}{m} \sum_{i=1,m} C_i$$

Where:

C_i = Center of ith active interval

m = Active length

5. STRAIGHTNESS

$$\text{Let } S = \sum_{i=3,m} \left| b_{i-2} - 2b_{i-1} + b_i \right| + \left| e_{i-2} - 2e_{i-1} + e_i \right|$$

Where:

b_i = Beginning of ith active interval

e_i = End of ith active interval

m = Active length

Then the straightness is:

$$1000 * \frac{S}{(m-2)}$$

6. EDGE DISCONTINUITY

$$S = \sum_{i=2,m} \left| W_i - W_{i-1} \right| \quad \text{Where } \left| W_i - W_{i-1} \right| > E$$

Where:

W_i = Width of ith active interval

E = Edge discontinuity threshold

Then the edge discontinuity is:

$$1000 * \frac{S}{(m-1)}$$

7. EDGE COUNT

$$\frac{1}{m} \sum_{i=1,m} E_i * 1000$$

Where:

E_i = Edge count for ith interval

m = Active length

8. BRIGHT COUNT

$$\frac{1}{A} \sum_{i=1,m} B_i * 1000$$

Where:

B_i = Bright count for ith interval

A = Active area

9. AVERAGE BACKGROUND

$$\frac{1}{m} \sum_{i=1,m} Z_i$$

Where:

Z_i = Value of the background estimator at the beginning of the ith interval

m = Active length

10. AVERAGE TARGET

$$\frac{1}{A} \sum_{i=1,m} T_i * W_i$$

Where:

T_i = Average target value in ith interval

W_i = Width of ith interval

A = Active area

11. PEAK TARGET INTENSITY

$\text{Max } \{P_i\}$

Where:

P_i = Peak intensity of the ith interval

12. FIRST LINE NUMBER

Line number of first interval

13. and 14. Beginning and ending column numbers for first interval

15. and 16. Interval endpoints for second interval

11 + 2 * I and

12 + 2 * I. Interval endpoints for ith interval (I ≤ 70)

151. and 152. Interval endpoints for 70th interval (if necessary)
(maximum of 70 intervals per bin)

153. to 240. Unused

241. LENGTH/AVERAGE WIDTH

$1000 * L/\overline{W}$

Where:

L = Total Length

$\overline{W}$ = Average Width

242. AVERAGE WIDTH

A/m

Where:

A = Active area

m = Number of active intervals (active length)

243. LEFT EDGE STRAIGHTNESS

Let $S = \sum_{i=1,m} \left(b_i - A_l \cdot L_i - B_l\right)$

Where:

b_i = Beginning of the ith interval (after median filtering)

A_l, B_l = Coefficients for least square linear fit for data points of the form $\left(L_i, b_i\right)$

L_i = Line number of the ith interval

m = Total length or 70, whichever is less (maximum of 70 intervals can be stored)

Then left edge straightness is:

$1000 * S/m$

244. RIGHT EDGE STRAIGHTNESS

Let $S = \sum_{i=1,m} \left(e_i - A_r \cdot L_i - B_r\right)$

Where:

e_i = End of ith interval after median filtering

A_r, B_r = (as in feature 243) for the interval endpoints

m = (as in feature 243)

Then right edge straightness is:

1000 * S/m

245. to 256. Unused

257. LEFT EDGE FEATURE

$$\frac{1}{m} \sum_{i=1,m} ES_i * 1000$$

Where:

m = Active length

ES_i = 1 if the ith active interval contains an edge at its left endpoint (edge at start), 0 if otherwise

m = Active length

258. RIGHT EDGE FEATURE

$$\frac{1}{m} \sum_{i=1,m} EE_i * 1000$$

Where:

m = The number of active intervals

EE_i = 1 if the ith active interval contains an edge at its right endpoint (edge at end)

m = Active area

259. to 300. Unused

Part 2, 74-Word Real Array

1. to 6. HARMONIC AMPLITUDES OF FOURIER BOUNDARY DESCRIPTORS

7. to 11. PHASE ANGLES OF THE FIRST FIVE FOURIER BOUNDARY DESCRIPTORS

12. to 14. FORM INVARIANT CROSS TERMS

Moment Features

Intensity	Silhouette	Boundary
15. μ_{00}	25. μ_{00}	35. μ_{00}
16. μ_{20}	26. μ_{20}	36. μ_{20}
17. μ_{11}	27. μ_{11}	37. μ_{11}
18. μ_{02}	28. μ_{02}	38. μ_{02}
19. μ_{30}	29. μ_{30}	39. μ_{30}
20. μ_{21}	30. μ_{21}	40. μ_{21}
21. μ_{12}	31. μ_{12}	41. μ_{12}
22. μ_{03}	32. μ_{03}	42. μ_{03}
23. $\bar{x}$	33. $\bar{x}$	43. $\bar{x}$
24. $\bar{y}$	34. $\bar{y}$	44. $\bar{y}$

$$\mu_{00} = \frac{1}{P} \sum_{m} \sum_{n} I(m,n)$$

Where:

m, n = A point within the object outline

P = The number of points in the summation

$$\bar{x} = \frac{\frac{1}{P} \sum_m \sum_n I(m,n) * m}{\mu_{00}}$$

$$\bar{y} = \frac{\frac{1}{P} \sum_m \sum_n I(m,n) * n}{\mu_{00}}$$

$$\mu'_{20} = \frac{1}{P} \sum_m \sum_n I(m,n) (m - \bar{x})^2$$

$$\mu'_{11} = \frac{1}{P} \sum_m \sum_n I(m,n) (m - \bar{x}) (n - \bar{y})$$

$$\mu'_{02} = \frac{1}{P} \sum_m \sum_n I(m,n) (n - \bar{y})^2$$

$$\mu'_{pq} = \frac{1}{P} \sum_m \sum_n I(m,n) (m - \bar{x})^p (n - \bar{y})^q$$

$$\mu'_{03} = \frac{1}{P} \sum_m \sum_n I(m,n) (n - \bar{y})^3$$

All moments are normalized with respect to $(\mu'_{02} + \mu'_{20})^{\frac{p+q}{2}}$ and μ_{00}.

Thus:

$$\mu_{20} = \mu'_{20} \Big/ \left[\left(\mu'_{02} + \mu'_{20} \right)^{\frac{2+0}{2}} \cdot \mu_{00} \right]$$

$$\mu_{11} = \mu'_{11} \Big/ \left[\left(\mu'_{02} + \mu'_{20} \right)^{\frac{1+1}{2}} \cdot \mu_{00} \right]$$

$$\mu_{02} = \mu'_{02} \Big/ \left[\left(\mu'_{02} + \mu'_{20} \right)^{\frac{0+2}{2}} \cdot \mu_{00} \right]$$

$$\mu_{pq} = \mu'_{pq} \Big/ \left[\left(\mu'_{02} + \mu'_{20} \right)^{\frac{p+q}{2}} \cdot \mu_{00} \right]$$

$$\mu_{03} = \mu'_{03} \Big/ \left[\left(\mu'_{02} + \mu'_{20} \right)^{\frac{0+3}{2}} \cdot \mu_{00} \right]$$

For the Intensity Moments:

I(m, n) = the intensity of the point
m, n in the original image

F r Silhouette Moments:

I(m, n) = 1 if m, n is in an object interval

I(m, n) = 0 otherwise

For Boundary Moments:

I(m, n) = 1 if

a. m, n is in an object interval and

b. one of the four orthogonal neighbors of m, n is not in an interval

I(m, n) = 0 otherwise

45. to 46. Unused

47. and 48. A_1 & B_1

Coefficients of the least square linear fit to the left endpoints

49. and 50. A_r & B_r

Coefficients of the least square linear fit to the right endpoints

51. θ_1

$\tan^{-1}(-1.0/A_1)$ in radians (The angle is adjusted so that $0 \leq \theta_1 \leq \pi$.)

52. θ_r

$\tan^{-1}(-1.0/A_r)$ in radians (The angle is adjusted so that $0 \leq \theta_r \leq \pi$.)

53. to 74. Unused

CLASSIFIER FORMAT

The format of the training tape and the format required by the classifier program (PATS3) consists of a 300-word integer array written and read with FORTRAN unformatted I/O. Each word consists of 32 bits. The contents of each word and scale factor are given in the following list:

Word	Contents	Scale
1	Not used	-
2	Bin number	-
3	Target type	-
4-10	Not used	-
11	Total length	-
12	Active length	-

Word	Contents		Scale
13	Total area		–
14	Center column		–
15	Straightness		–
16	Edge discontinuity		–
17	Edge count		–
18	Bright count		–
19	Average background intensity		–
20	Average target intensity		–
21	Peak target intensity		–
22	First line number		–
23-162	Interval columns		–
163	A_o	Fourier boundary Descriptors Amplitude	S1
164	A_1		S2
165	A_2		S2
166	A_3		S2
167	A_4		S2
168	A_5		S2
169	ϕ_1	Fourier boundary Descriptors Phase	S4
170	ϕ_2		S4
171	ϕ_3		S4
172	ϕ_4		S4
173	ϕ_5		S4
174	F21	Fourier boundary Descriptors Cross terms	S4
175	F32		S4
176	F31		S4

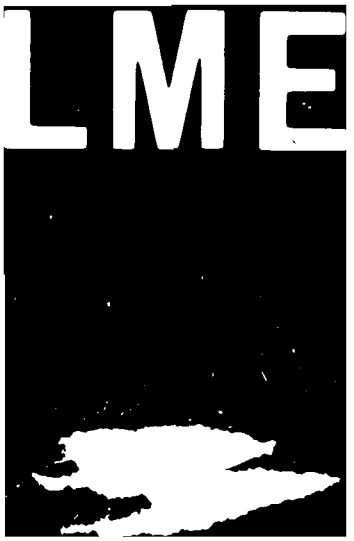
Word	Contents		Scale
177	μ_{00}		S3
178	μ_{20}		S5
179	μ_{11}		S5
180	μ_{02}	Intensity	S5
181	μ_{30}	moments	S5
182	μ_{21}		S5
183	μ_{12}		S5
184	μ_{03}		S5
185	$\bar{x}$		S2
186	$\bar{y}$		S2
187	μ_{00}		S5
188	μ_{20}		S5
189	μ_{11}	Silhouette	S5
190	μ_{02}	moments	S5
191	μ_{30}		S5
192	μ_{21}		S5
193	μ_{12}		S5
194	μ_{03}		S5
195	$\bar{x}$		S2
196	$\bar{y}$		S2
197	μ_{00}		S5
198	μ_{20}	Boundary	S5
199	μ_{11}	moments	S5
200	μ_{02}		S5
201	μ_{30}		S5

Word	Contents		Scale
202	μ_{21}	Boundary moments	S5
203	μ_{12}		S5
204	μ_{03}		S5
205	$\bar{x}$		S2
206	$\bar{y}$		S2
207	Length/average width		
208	Average width		
209	Left edge straightness		
210	Right edge straightness		
211	A_{left}		S4
212	B_{left}		S2
213	A_{right}		S4
214	B_{right}		S2
215	θ_{left}		S4
216	θ_{right}		S4
217	Minimum target intensity		–
218	Peak target intensity		–
219	Average target contrast		–
220	Average absolute contrast		–
221	Perimeter / $\sqrt{\text{area}}$		S4
222	1: Bright		–
	-1: Cold		–
223-224	Not used		–

Word	Contents	Scale
225	Left edge feature	-
226	Right edge feature	-
227-300	Not used	-

The calculation of these features is described in the BINS TAPE section. The following scale values have been used in order to preserve the significance of the features when converting to integer format:

Scale	Multiplier
S1	300
S2	1,000
S3	10,000
S4	100,000
S5	1,000,000


LME